MinervaDB

Building MySQL Infrastructure for Performance and Reliability

About me

Name - Shiv Iyer

Occupation - Founder @ MinervaDB, Database Systems Architect

Technology Focus:

- Open Source Database Systems
- MySQL
- MariaDB
- PostgreSQL
- ClickHouse
- Database Systems Performance, Scalability and Data SRE

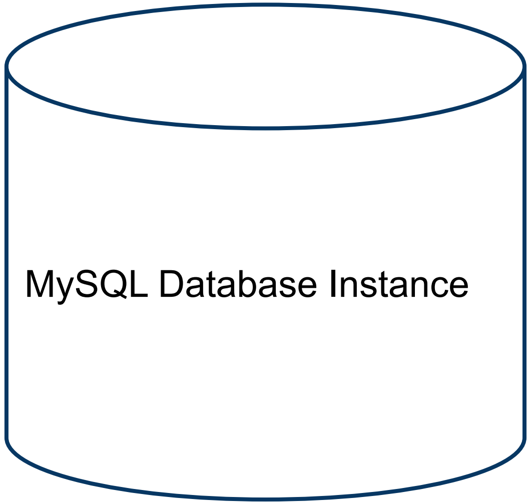
MinervaDB

Interesting things to know

- Database Infrastructure Operations get complex as your business grows, Fast growing companies Database Systems are built to scale proactively.
- Do not plan to scale your Database Infrastructure operations by eyeballs - Automate and hire Data Ops. geeks who are specialists in Database Systems Operations.
- Cost of Database Systems outage for a business:
  - Data Infrastructure outage impacts entire revenue ecosystem and customer loyalty - Cost to Customer Acquisition is significantly high many SaaS business
  - Extended work hours for your employees - Higher attrition is no good

MinervaDB

This is how it begins for everyone ...



- Single database instance for both READS and WRITES
- Vertically scaled and optimized
- Many MySQL system variables are tuned reactively
- Indexes are created very generously
- Everyone is a DBA here
- Backups are mostly not audited or there is absolutely no infrastructure to audit backups regularly
- I call this setup an optimistic Database Infrastructure Operations management - Strong belief from everyone that nothing will ever fail

Everything changes with time

Remember that changes are a normal part of Database Systems operations, It is also amazing to see how Database Infrastructure grow from silo to distributed and redundant platforms addressing performance, scalability, high availability, fault-tolerant, self-healing, reliable and secured transaction processing engines..

“The only constant in life is change”-Heraclitus

Business growth and strategic thoughts

- Single-point-of-failure outage (human error or Infrastructure crash)
- Performance - Too much READs and WRITEs going on a single Database Systems Infrastructure
- More data is operationally expensive on a single MySQL instance
- Need a dedicated and accountable Database Infrastructure Operations expert
- Let's build Database Systems operations for growth
- Reliability is above performance as we grow

MinervaDB

What you expect from Database Infra.?

- Optimal performance using System Resources efficiently
- Scalable - Vertically and Horizontally
- Robust DR setup
  - High performance backup and restore
  - Backup retention in remote DC / site and cloud
  - Infrastructure for regular auditing of MySQL backup
- Highly Available, Distributed, Redundant, Fault-Tolerant, Self-Healing and Secured
- **Reliability over performance**
- Database Infrastructure Operations visibility - Observability and Resilience

MinervaDB

High Availability and "Nines SLAs"

Build your Database Infrastructure for Reliability !

Availability %	Downtime Per Year
90% ("one 9")	36.53 days
99% ("two 9s")	3.65 days
99.9% ("three 9s")	8.77 hours
99.99% ("four 9s")	52.60 minutes
99.999% ("five 9s")	5.26 minutes
99.9999$ ("six 9s")	31.56 seconds

MinervaDB

Performance of Database Systems Ops.

- Tuning Linux for MySQL performance
- Tuning MySQL for performance
- Cost efficient SQL and efficient indexing (more is not good for indexing strategy)
- Optimal disk I/O, partitioning and archiving for your MySQL infrastructure
- Replication for MySQL Performance and Scalability
- Choosing MySQL storage engines for performance
- Time-series Performance Monitoring Infrastructure for MySQL operations

Tuning Linux for MySQL Performance

- Invest more on RAM for MySQL Server transaction performance and configure Linux to do minimal swapping / reduce the I/O operations is the goal

```
# Set the swappiness value as root
echo 1 > /proc/sys/vm/swappiness

# Verify the change
cat /proc/sys/vm/swappiness
1
```

We don't recommend value "0" which disables swapping completely and value 1 does a very minimal swapping. The configuration can be persisted in /etc/sysctl.conf:

```
vm.swappiness = 1
```

MinervaDB

Tuning MySQL Server for Performance

Rule 1 - Over tuning is definitely no good for MySQL performance, We are tuning MySQL Server mostly for I/O Ops. velocity and there is no magic MySQL system variables exist to make your queries faster. OPTIMAL QUERIES AND EFFICIENT INDEXING MAKES MYSQL FASTER.

Rule 2 - Tune MySQL System Variables which can definitely improve your Database Infrastructure Operation performance and it's strongly recommended to tune only one parameter at a time to avoid serious mistakes / confusions.

MinervaDB

What I look for immediately after MySQL installation

- innodb_buffer_pool_size - 80% of total RAM
- innodb_flush_log_at_trx_commit - 1 for MySQL Master and 0 / 2 (less reliable) for MySQL Slaves
- innodb_log_file_size - Not more than 2GB for faster MTTR
- innodb_flush_method - O_DIRECT to avoid double buffering

Recommended read - MySQL 5.7 Performance Tuning Immediately After Installation

MinervaDB

MySQL System Variables which allocate Memory per connection / thread

- sort_buffer_size
- read_rnd_buffer_size
- join_buffer_size
- read_buffer_size
- tmp_table_size

NOTE - Setting these MySQL System Variables very high causes excessive swapping and in extreme cases MySQL Server will crash continuously

MinervaDB

Please do not tune sort_buffer_size

- MySQL system variable *sort_buffer_size* is a per session buffer, i.e. memory is assigned per connection / thread. So to conclude large *sort_buffer_size* is definitely a serious problem.

  Recommendation - I never bother to change this value, There are several books / posts which advise you to closely monitor MySQL status variable *sort_merge_passes* and if it increases then tune *sort_buffer_size* , In reality it just damage your MySQL performance.

Configuring InnoDB for Disk I/O Performance

Configuring InnoDB System Variables when using non-rotational storage devices

- innodb_checksum_algorithm - crc32 uses a faster checksum algorithm and is recommended for faster storage systems
- Disable innodb_flush_neighbors for performance
- You can carefully increase innodb_io_capacity for performance (default - 200) after successful benchmarking
- You can carefully increase innodb_io_capacity_max for performance (default - 2000) after successful benchmarking

Configuring InnoDB System Variables when using non-rotational storage devices

- Disable innodb_log_compressed_pages to reduce the logging
- Configure innodb_page_size very close to storage device block size
- Optimally configure innodb_log_file_size to maximize caching and write combining
- Configure binlog_row_image to "minimal" to reduce logging
- Increase innodb_io_capacity to avoid backlogs which causes bottleneck to the throughput

MinervaDB

Configuring InnoDB System Variables when using non-rotational storage devices

- By default innodb_log_compressed_pages are enabled to prevent the corruption that can occur if a different version of zlib compression algorithm is used during recovery, If you are confident zlib version will not change then disable innodb_log_compressed_pages to reduce redo log generation for workloads that modify compressed data.

How to measure MySQL performance ?

- Response Time - The business care about only **Response Time** of the transaction, Cache Hit Ratio makes no sense for them at all
- **Why bother about throughput then ?**
  - High performance CPU, Memory and Disk / SSDs does matter for capacity planning / sizing of transaction performance
  - Building MySQL infrastructure for performance:
    - Sequential I/O ops. files - Binary logs, REDO logs (ib_logfile*), slow query log, audit log and error log (HDD with battery-backed write-cache)
    - Random I/O ops. files - innodb data files, SSDs or NVRAM card, or high RPM spindle disks like SAS 15K or 10K, with hardware RAID controller and battery-backed unit.

Linux utilities to troubleshoot the performance

Monitoring Linux process using "top"

- Process ID
- CPU usage
- Memory usage
- Swap memory
- Cache size
- Buffer size
- User commands

Linux utilities to troubleshoot the performance

VmStat

- Used to monitor virtual memory statistics
- VmStat is shipped with Sysstat package
- You can monitor following statistics from VmStat:
  - System processes
  - CPU activity
  - Virtual memory
  - Disk operations
  - Kernel threads
  - I/O blocks
  - Interrupts

MinervaDB

Linux utilities to troubleshoot the performance

Lsof

- Monitor the list of open files and their related process
- This command is used usually when you have to monitor which files are in use and the disk cannot be unmounted due to the error files are opened
- The following statistics are displayed with Lsof command:
    - Processes
    - devices
    - disk files
    - Network sockets
    - Pipes

MinervaDB

Linux utilities to troubleshoot the performance

Iotop

- The real-time disk I/O statistics and processes can be monitored with Iotop:
    - Processes
    - User
    - Disk Reads
    - Disk Writes
    - I/O %
    - Command used

MinervaDB

Linux utilities to troubleshoot the performance

IoStat

- Monitor input and output statistics of storage devices
- The following statistics are delivered from IoStat command:
    - Devices details / information
    - TPS
    - Blocks read/second
    - Blocks written/second
    - Total blocks read
    - Total blocks written

MinervaDB

Troubleshooting MySQL query performance

Configuring MySQL System Variables for troubleshooting query performance:

- slow_query_log #log slow queries
- long_query_time #log queries which takes more than "X" seconds
- log_queries_not_using_indexes # log queries not using indexes
- log_slave_slow_statements # log slow queries in replication slave

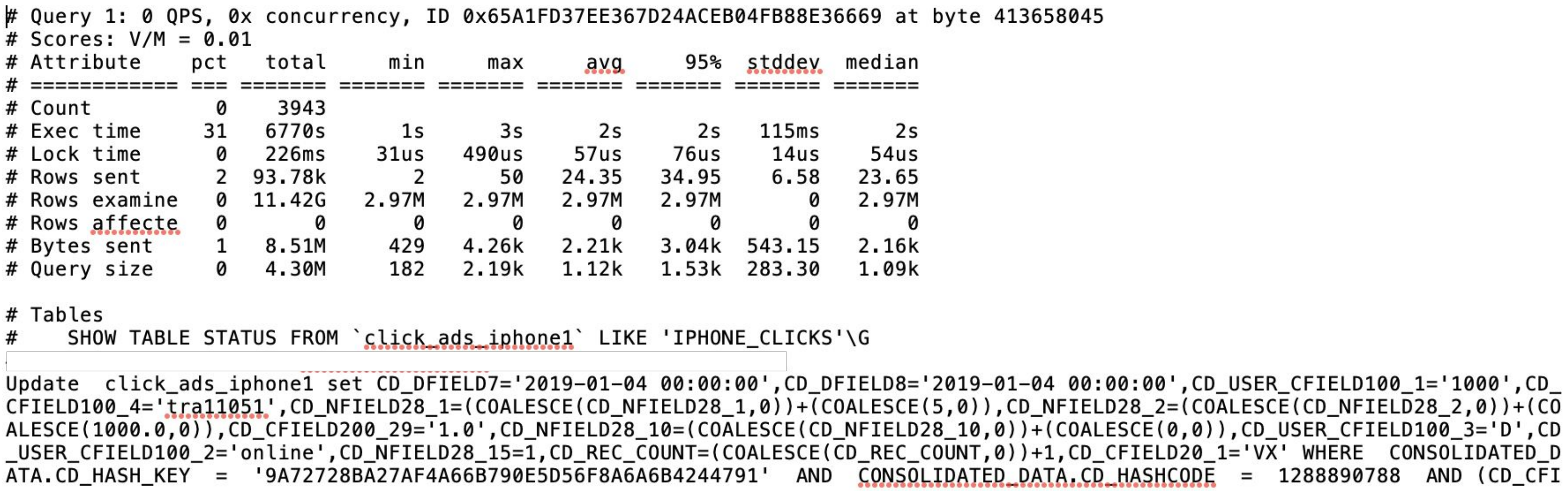
P.S. - Slow Queries are only added to the slave's slow query log only when they are logged in statement format in the binary log, i.e. when binlog_format=STATEMENT is set, or when binlog_format=MIXED is set and the statement is logged in statement format.

Troubleshooting MySQL Performance with Slow Query Log

- **Slow Query Log** consist of queries that took more than **long_query_time** in seconds to execute.
- Most popular tools to annotate MySQL slow query log:
  - mysqldumpslow
  - pt-query-digest (**Percona Toolkit**)
- Orchestrating MySQL slow query log parameters for higher visibility
  - To log expensive MySQL administrative statements enable log_slow_admin_statements
  - To log those queries which are not using indexes enable **log_queries_not_using_indexes**

MinervaDB

Slow Query log interpreted with pt-query-digest



```
# Query 1: 0 QPS, 0x concurrency, ID 0x65A1FD37EE367D24ACEB04FB88E36669 at byte 413658045
# Scores: V/M = 0.01
# Attribute    pct   total     min     max     avg     95%  stddev  median
# ============ === ======= ======= ======= ======= ======= ======= =======
# Count          0    3943
# Exec time     31   6770s      1s      3s      2s      2s   115ms      2s
# Lock time      0   226ms    31us   490us    57us    76us    14us    54us
# Rows sent      2  93.78k       2      50   24.35   34.95    6.58   23.65
# Rows examine   0  11.42G   2.97M   2.97M   2.97M   2.97M       0   2.97M
# Rows affecte   0       0       0       0       0       0       0       0
# Bytes sent     1   8.51M     429   4.26k   2.21k   3.04k  543.15   2.16k
# Query size     0   4.30M     182   2.19k   1.12k   1.53k  283.30   1.09k

# Tables
#    SHOW TABLE STATUS FROM `click_ads_iphone1` LIKE 'IPHONE_CLICKS'\G

Update  click_ads_iphone1 set CD_DFIELD7='2019-01-04 00:00:00',CD_DFIELD8='2019-01-04 00:00:00',CD_USER_CFIELD100_1='1000',CD_
CFIELD100_4='tra11051',CD_NFIELD28_1=(COALESCE(CD_NFIELD28_1,0))+(COALESCE(5,0)),CD_NFIELD28_2=(COALESCE(CD_NFIELD28_2,0))+(CO
ALESCE(1000.0,0)),CD_CFIELD200_29='1.0',CD_NFIELD28_10=(COALESCE(CD_NFIELD28_10,0))+(COALESCE(0,0)),CD_USER_CFIELD100_3='D',CD
_USER_CFIELD100_2='online',CD_NFIELD28_15=1,CD_REC_COUNT=(COALESCE(CD_REC_COUNT,0))+1,CD_CFIELD20_1='VX' WHERE  CONSOLIDATED_D
ATA.CD_HASH_KEY  =  '9A72728BA27AF4A66B790E5D56F8A6A6B4244791'  AND  CONSOLIDATED_DATA.CD_HASHCODE  =  1288890788  AND (CD_CFI
```

MinervaDB

Interpreting **Slow Query Log**

- **Query_time** - The SQL execution duration in seconds
- **Lock_time** - The time to acquire locks in seconds
- **Rows_sent** - Number of rows sent to client
- **Rows_examined** - Number of rows examined by the optimizer

Handling MySQL Growth in Terabytes

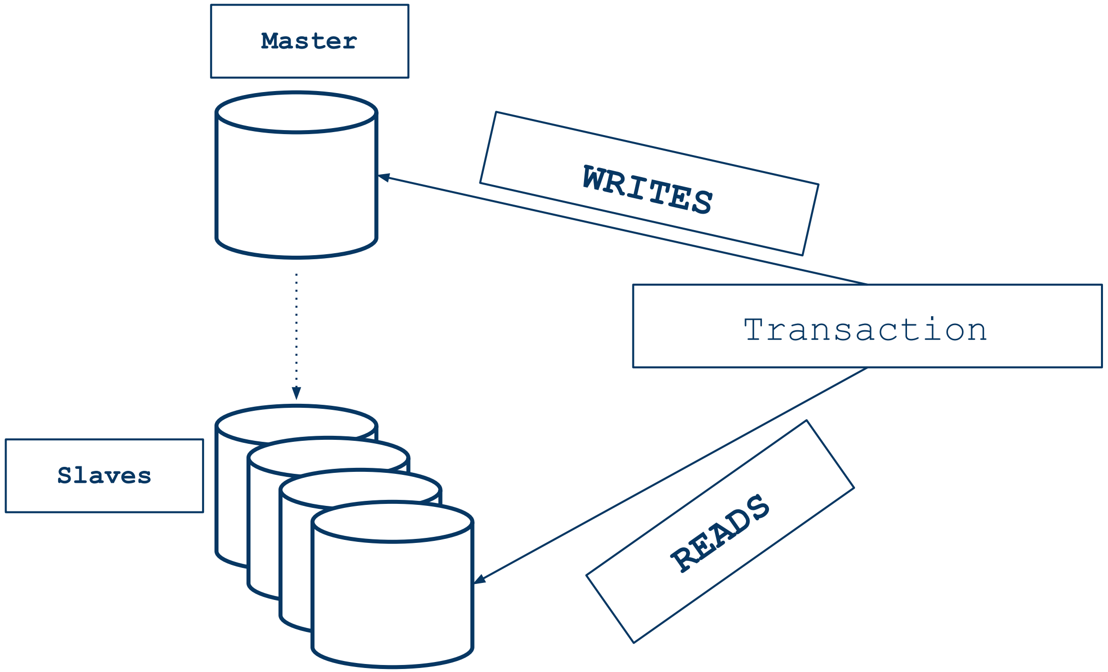
- Identify transaction intensive Tables proactively to strategize MySQL Database growth plans
- Large tables are operationally expensive - Query / Index performance, Scalability and Backup & Restoration
- Avoid single-point-of-failure on large tables:
    - Building Database Systems infrastructure distributed, redundant, fault-tolerant and self-healing:
        - Replication for scalability and HA
            - Asynchronous Replication
            - Synchronous Replication
        - Distribute READs and WRITEs across nodes via load-balancers in MySQL Replication infrastructure for Database Ops. reliability

MySQL replication topologies

- Scale-out
  - Standard READ-ONLY Slaves to split READs exclusively
    - Slaves for running super expensive sub-optimal queries
    - Slaves dedicated to run SORT / SEARCH intensive SQLs
      - Aggregation SQLs
    - Slaves for fulltext searches
    - Slaves for warehouse queries
- Building distributed Data Ops. for WebScale
  - Slaves for Data Archiving (**Horizontal Shards**)
  - Slaves for DR / Backup
  - Delayed Slaves for Rollback

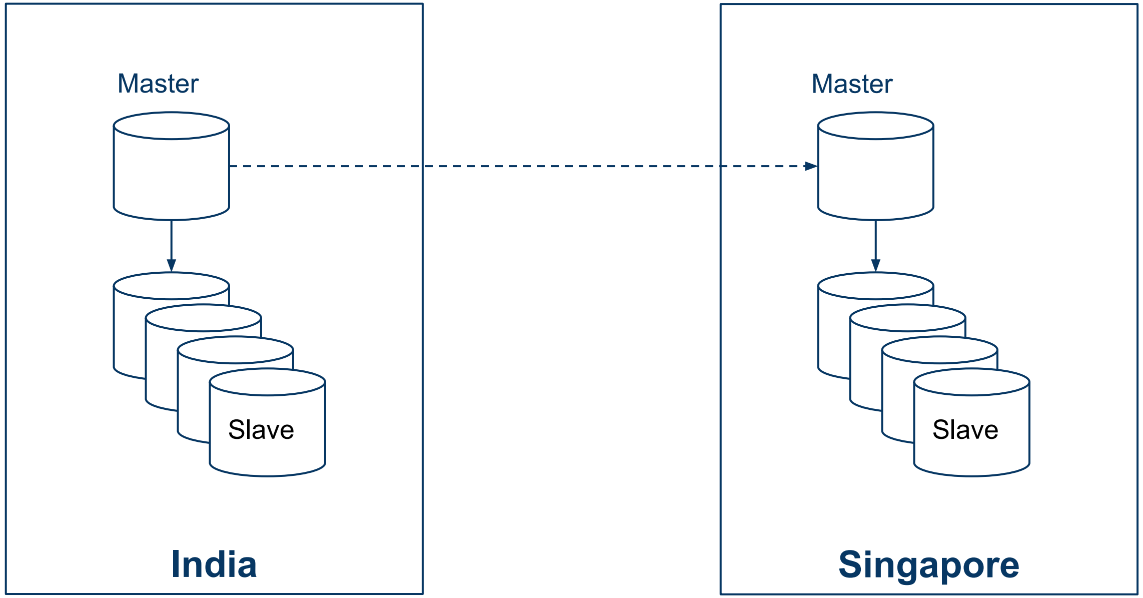
MinervaDB

MySQL Replication to Scale-out



MinervaDB

Distributed Relay Slave Replication

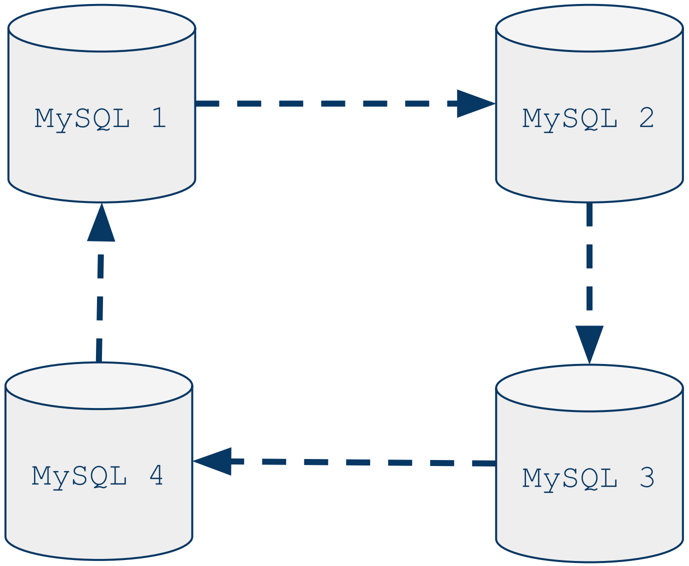


MinervaDB

MySQL Ring Replication

- Master can have many slaves
- Slave can have only one master
- Log_slave_updates and server_id is configured in MySQL Ring Replication

MySQL Ring Replication



MinervaDB

Why I don't recommend Ring Replication?

- Super sensitive and not reliable
  - Every member of the ring has its own binlog position
  - #MySQL 4 fails, binlog position is lost

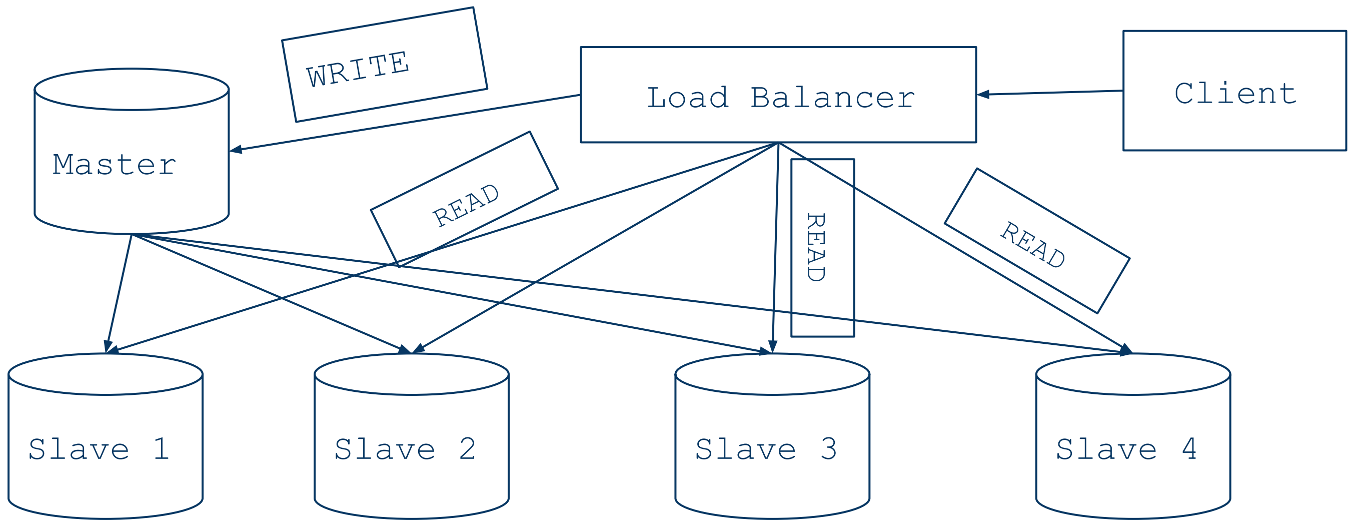
P.S. - Highly compelling multi-master replication topology, Many DBAs in the past had been the victim of first impression

How to use MySQL replication for performance ?

- Splitting READs and WRITEs
  - Most of websites READ intensive, The users are either reading blogs / posts or checking offers of products / services to buy. The UPDATE happens only when they comment or buy specific article / service.
  - Use slaves for warehouse and analytics queries

MinervaDB

MySQL Replication for Performance



MinervaDB

Why attempting scaling WRITEs is not worth ?

If you can scale WRITEs successfully then you have taken care of the biggest bottleneck of building web-scale database infrastructure operations. But, How much true is this ? Imagine you are solving this with an clustering solution which splits WRITEs across two instances W1 and W2. In reality, you are repeating writes on both machines. Yes, All WRITEs on W1 will be repeated on W2 and all WRITEs on W2 will be repeated on W1. To make this solution even more scary, When there is a database outage both instance will not have reliable data, So how can scaling WRITEs really worth ? I don't agree !

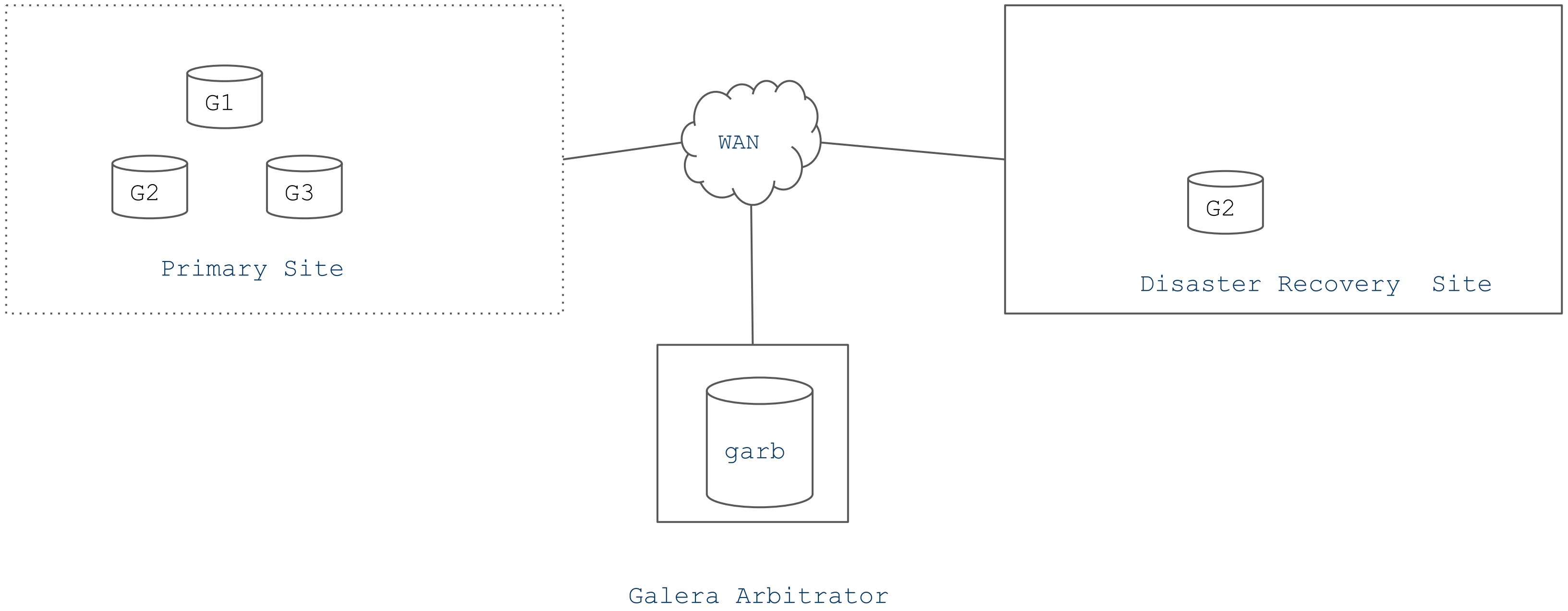
MinervaDB

Using MySQL replication for disaster recovery

- MySQL replication is an high availability solution from MySQL, HA cannot be supplemented with DR (Disaster Recovery)
- How to use MySQL replication node for DR?
  - Logical / cold backup of MySQL database replication node (master / slave) using mysqldump (always use **--lock-all-tables**)
  - Percona XtraBackup - Hot backup solution from Percona Server
  - Zero data loss DR solution using Galera Cluster

Zero data loss DR solution using Galera Cluster

ACTIVE PASSIVE MASTER - MASTER CLUSTER



MinervaDB

ACTIVE PASSIVE MASTER - MASTER CLUSTER

Failover strategy - Redirect data traffic totally to DR site if there is an outage

Fallback strategy - Redirect data traffic back to primary site

Advantages - Failover and Fallback without any downtime

Disadvantages - Expensive, Failover site will be idle most of time (Oversized infra.) and low / reliable latency network infrastructure to avoid the performance bottlenecks

MinervaDB

Data Ops. geek checklist for success

- Reliable MySQL Support
- Observability & Resilience Infrastructure- Monitoring MySQL performance proactively
- Robust backup strategy
- Reliable and self-healing Database Infrastructure Operations
- Database security and privacy
- Invest in education and new skills - Attend conferences, webinars, follow blogs and books

How can you contact me ?

Email - shiv@minervadb.com

Twitter - https://twitter.com/thewebscaledba

LinkedIn - https://www.linkedin.com/in/thewebscaledba/

Phone / MinervaDB Toll Free - (844) 588-7287